Received February 9, 2017, accepted March 21, 2017, date of publication April 5, 2017, date of current version April 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2688477

An Effective Algorithm Based on Density Clustering Framework

JIANYUN LU[1,2], (Member, IEEE), AND QINGSHENG ZHU[1,3], (Member, IEEE)

[1]College of Computer Science, Chongqing University, Chongqing 400044, China
[2]Chongqing College of Electronic and Engineering, Chongqing 401331, China
[3]Chongqing Key Laboratory of Software Theory and Technology, College of Computer Science, Chongqing University, Chongqing 400044, China

Corresponding author: Q. Zhu (qszhu@cqu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61272194.

ABSTRACT Clustering analysis has the very broad applications on data analysis, such as data mining, machine learning, and information retrieval. In practice, most of clustering algorithms suffer from the effects of noises, different densities and shapes, cluster overlaps, etc. To solve the problems, in this paper, we propose a simple but effective density-based clustering framework (DCF) and implement a clustering algorithm based on DCF. In DCF, a raw data set is partitioned into core points and non-core points by a neighborhood density estimation model, and then the core points are clustered first, because they usually represent the center or dense region of the cluster structure. Finally, DCF classifies the non-core points into initial clusters in sequence. In experiments, we compare our algorithm with Dp and DBSCAN algorithms on synthetic and real-world data sets. The experimental results show that the performance of the proposed clustering algorithm is comparable with DBSCAN and Dp algorithms.

INDEX TERMS Clustering algorithms, reverse k-nearest neighbors, neighborhood density estimation, data mining, minimum spanning tree.

I. INTRODUCTION

Clustering studies play an important role in many scenarios of knowledge discovery in databases (KDD), including data mining, information retrieval and image segmentation. The goal of clustering algorithms is to partition the dataset into clusters so that the similarities of objects in the same cluster are maximized and the similarities of objects between different clusters are minimized [1].

In practice, the clustering results usually suffer from the effects of dataset characteristics (e.g. noises, overlaps, different shapes, densities and sizes) and the limitations of clustering algorithms (such as many parameters). Thus, it is hard to find a comprehensive clustering algorithm to cope with all the situations. For example, partitioning-based clustering method k-means [2] is fragile to noises because the noises make the mean value of the cluster change greatly. Another example is density-based method DBSCAN [3] which is robust to noises, shapes and densities and can determine the number of clusters automatically, however, it has two parameters (*MinPts*, *Eps*) that require user to set in practical applications. Furthermore, DBSCAN is weak to distinguish the significant overlaps between clusters and sensitive to the variation of density in the same cluster. Recently, Alex and Alessandro proposed a fast density-based clustering algorithm named Dp to find cluster centers [4]. In Dp algorithm, the measure of local density for points is like to DBSCAN, which uses a distance cutoff parameter d_c to compute the number of points located in the d_c-neighborhood. Besides, it also needs parameter *MinPts* in the practice. Note that it is not always reliable and stable to use one point to represent the cluster center. In our view, it is more reasonable to utilize core points to represent the characteristics of a cluster.

To improve the robustness of clustering algorithm in practice, in this paper we propose a **D**ensity-based **C**lustering **F**ramework (DCF) and implement a clustering algorithm based on DCF to validate the clustering results. In DCF, a raw dataset is firstly partitioned into core points and non-core points by a neighborhood density estimation model. Then initial clustering operation is carried out for the core points which usually represent the center or dense region of clusters. Finally, non-core points are classified into initial clusters in a priority sequence. We evaluate our method on synthetic and real-world datasets. The experimental results show that the proposed algorithm achieves comparable performance with DBSCAN and Dp algorithms.

The remainder of the paper is organized as follows. In Section II, we briefly review the related work of clustering analysis. In Section III, we give the density-based clustering framework and discuss its work process in detail. A clustering algorithm based on DCF is developed in Section IV. Section V shows the reports of the experiments. The paper is concluded in Section VI.

II. RELATED WORK

Clustering analysis has a wide application fields such as data mining, image segmentation and community network. For several decades, many of clustering algorithms have been proposed. According to different methodology, these clustering algorithms are loosely divided into five categories: partitioning-based, hierarchical-based, density-based, graph-based and model-based algorithms [5].

Partitioning-based clustering algorithms partition a dataset into several clusters by the distance from the point to the cluster center but the number of output clusters needs expert or user to specify. The advantage of this kind of clustering algorithms is that it is simple and efficient. However, the clustering results are susceptible to noises and non-convex shape of data distribution. Typical algorithms of this category are k-means and its variants [6], [7]. Fuzzy c-means [8] is one of fuzzy clustering algorithms that can assign a data point to all clusters with a certain degree of membership.

Hierarchical-based clustering algorithms usually obtain a clustering structure of the dataset called dendrogram which is a hierarchical tree structure showing the inner links of objects. The dendrogram can be created by agglomerating (considering each object as a cluster) or dividing (considering the whole dataset as a cluster). On the basis of linkage criterion, there are single-link and complete-link hierarchical clustering algorithms [9]. The drawback of this kind of algorithms is that it is computationally prohibitive when they construct a dendrogram for a large dataset. Xu, et al. proposed a density peak based hierarchical clustering method (DenPEHC), which generates clusters directly on each possible clustering layer, and introduces a grid granulation framework to enable DenPEHC to cluster large-scale and high-dimensional (LSHD) datasets [10].

DBSCAN and Dp algorithms usually use a function model to estimate the local density for each object in a dataset. DBSCAN classifies objects into three types (i.e., core objects, border objects and outliers) by the number of neighbors contained in the *Eps* neighborhood. Besides, the NBC [11] algorithm uses the ratio of reverse k-nearest neighbors to k-nearest neighbors to measure objects' local density. However, DBSCAN or NBC cannot detect the significant overlaps between clusters and makes a mistake in clustering. Recently, a fast density-based clustering algorithm called Dp is proposed in [4], which uses d_c-neighborhood to estimate local density ρ for points and takes δ as a function of ρ to find cluster center. Zhu, et al. proposed a density-ratio clustering algorithm, which can identify and analyze the condition under which the density-based clustering algorithms fail [12]. But ReCon-DBSCAN algorithm needs one additional parameter (η) in DBSCAN and ReScale-DBSCAN needs two additional parameters (η and ψ).

Graph-based clustering algorithms partition data into clusters based on a graph (such as kNN graph, MST graph). In the graph, vertexes and the weights of edges represent objects and similarities between objects respectively. For example, MST-based [13] clustering algorithms find the $k-1$ largest weights in the MST graph and remove them to form k clusters. The CHAMELEON clustering algorithm [14] is based on kNN graph, which firstly splits the kNN graph into sub-graphs, and then merges sub-graphs to build a hierarchical structure according to a dynamic modelling. Gan, et al. proposed a graph-based clustering algorithm called "probability propagation", which can identify clusters with spherical shapes and non-spherical shapes [15].

Model-based clustering algorithms assume sample observations arise from a distribution that is a mixture of two or more components and each component in the mixture is a cluster. These models include Gaussian model [16], Latent Dirichlet Allocation [17], etc. The most used expectation-maximization (EM) algorithm [18] is to infer the parameters in a mixture of Gaussian models. However, EM algorithm suffers from the slow convergence rate and the possibility of a local optimum. The major disadvantages of model-based clustering algorithms are that they are sensitive to the selection of model parameters at first and the number of clusters.

III. CLUSTERING FRAMEWORK

As we know, clustering algorithms usually suffer from the effects of noises, cluster overlaps, different densities, sizes and shapes. Besides, none of the available clustering algorithms can solve all the problems in practice. The trend of cluster analysis is combining the advantages of different clustering algorithms to achieve desired results in engineering application [19]. For this purpose, we propose a density-based clustering framework (DCF) which can combine different clustering algorithms to obtain better clustering results. We show the internal structures of the DCF in Fig.1. The DCF mainly contains four modules: density partition, initial clustering, ordering and partition clustering. For clarity, they are pointed out by A, B, C and D in Fig.1.

We introduce the workflow of DCF as follows: 1) Partitioning a dataset into core points and non-core points by a neighborhood density estimation model. 2) Clustering the core points to obtain the initial clusters. 3) Building a priority sequence for non-core points. 4) Classifying the non-core points into the initial clusters in sequence. Next, we will describe each module.

A. DENSITY PARTITION

In this module, the aim is to partition the D into core points and non-core points using neighborhood density estimation. In practice, non-core points (e.g. noises and border points) usually affect the clustering results. Therefore, it is necessary to remove them temporarily before clustering. So far, neighborhood density estimation is an effective measure to accomplish this goal such as the way of that used in *LOF* [20], *INFLO* [21], *IP* [22], etc. So, the D is not difficult to be

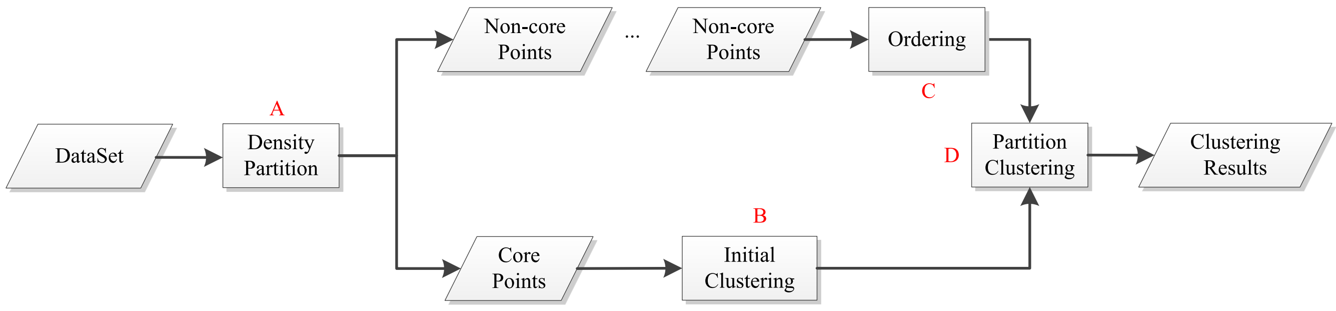


FIGURE 1. The density-based clustering framework.

partitioned into two sets through a neighborhood density threshold. One set is core points with higher neighborhood density and the other set contains non-core points with lower neighborhood density. For a large dataset or serious overlaps, we can execute density partition more times to find out core points.

B. INITIAL CLUSTERING

After the operation of density partition, the dataset D will be divided into several sets. However, among them, only one set includes core points and the remaining sets contain non-core points. Intuitively, the core points which represent cluster structures are separated distinctly. If we use clustering algorithm on the core points, it will achieve better clustering results. In this module, we can apply K-means, DBSCAN or MST clustering algorithms to obtain the initial clusters because these algorithms are simple and effective.

C. ORDERING

In fact, how to classify non-core points into initial clusters has an influence on the final cluster shape. Intuitively, we should classify the non-core points into initial clusters in a priority sequence. This module provides a sorting mechanism for non-core points. A simple way to build a priority sequence is to use neighborhood density of non-core points.

D. PARTITION CLUSTERING

To obtain the final clustering results, the last step is to classify the non-core points into the initial clusters in sequence. This module needs design a function to measure the similarity between the non-core points and the initial clusters. Certainly, we can use some existed similarity measures such as nearest neighbor, shared nearest neighbors [23] or other classifier algorithms.

IV. A CLUSTERING ALGORITHM BASED ON DCF

In this section, we implement a clustering algorithm based on the DCF to validate the clustering results.

A. NEIGHBORHOOD DENSITY ESTIMATION

We use reverse k-nearest neighborhood (RkNN) [24] as the neighborhood density estimation model. RkNN is not only simple and fast for computing, but also effective to evaluate the local density. It is successfully used in hubs and outliers detection. The definition of RkNN is given as follows.

Definition 1 (Reverse k-Nearest Neighborhood of a Point p): In a data set D and given k, reverse k-nearest neighborhood of a point p is defined as:

$$RkNN(p) = \{q | p \in kNN(q), q \neq p, q \in D\} \tag{1}$$

In equation (1), where kNN(q) is k-nearest neighborhood of the point q. If the point p has a big size of RkNN, it indicates that p appears more in other points' k-nearest neighborhood and it is likely a core point, and vice versa. We give the definition of neighborhood density by normalizing the number of RkNN.

Definition 2 (Neighborhood Density of a Point q): In a dataset D and given k, the neighborhood density (ND) of a point q is defined as:

$$ND(q) = \frac{|RkNN(q)| - \min(|RkNN(o)|)}{\max(|RkNN(o)|) - \min(|RkNN(o)|)}, \quad q, o \in D \tag{2}$$

Definition 3 (Density Partition of a Dataset D): Given a dataset D, a density partition of D with a neighborhood density threshold T satisfying that:

$$\begin{cases} D = CP_{ND(p)>T}(p) \cup NCP_{ND(q)\leq T}(q), \\ \qquad p, q \in D, \quad T \in (0, 1) \\ \emptyset = CP_{ND(p)>T}(p) \cap NCP_{ND(q)\leq T}(q), \\ \qquad p, q \in D, \quad T \in (0, 1) \end{cases} \tag{3}$$

In equation (3), where $CP_{ND(p)>T}(p)$ is a set including core points with higher neighborhood density than the threshold T, $NCP_{ND(q)\leq T}(q)$ contains non-core points whose neighborhood densities are lower than or equal to the threshold T. For a large dataset or clusters with serious overlaps, we can repeat density partition on the core points set. The detailed density partition algorithm is showed in Fig.2.

B. MINIMUM SPANNING TREE CLUSTERING

Intuitively, the core points are dense regions of clusters and usually represent cluster structures. We apply minimum spanning tree (MST) algorithm to the core points to obtain the initial clusters because MST can discover clusters with different shapes and sizes. MST is a subset of the edges of

```
Algorithm Density partition
Input: A dataset $D$ with $N$ points, $k$ the number of neighbors
Output: $CP$, $NCP$, $ND_{NCP}$, $IS_{NCP}$ /* $CP$ is a core point set, $NCP$ is a non-core point set, $ND_{NCP}$ is an array containing neighborhood density of non-core points, $IS_{NCP}$ is a matrix containing influence space of non-core points */
Begin
1. Initializing $k$, $CP=\emptyset$, $NCP=\emptyset$;
2. Building $k$-$d$ tree for $D$;
3. parfor each point $p\in D$ do /* parallel computing kNN for each point */
4.     Finding $k$NN($p$);
5. end parfor
6. for each point $p\in D$ do
7.     Computing the size of R$k$NN($p$) set;
8. end for
9. for each object $p\in D$ do
10.    Computing $ND(p)$; /* according to definition 2 */
11. end for
12. for each object $p\in D$ do
13.    if $ND(p)>T$; /* $T$ is a neighborhood density threshold, $T\in(0,1)$ */
14.        $CP=CP\cup\{p\}$;
15.    else
16.        $NCP=NCP\cup\{p\}$;
17.        $ND_{NCP}(p)=ND(p)$;
18.        $IS_{NCP}(p)=k\text{NN}(p)\cup\text{R}k\text{NN}(p)$;
19.    end
20. end for
21. Returning $CP$, $NCP$, $ND_{NCP}$, $IS_{NCP}$;
End.
```

FIGURE 2. **The density partition algorithm.**

```
Algorithm Initial clustering
Input: $CP$ the core point set, $c$ the number of clusters
Output: $c$ clusters
Begin
1. Initializing $c$;
2. Computing Euclidean distance Matrix $EM$ for $CP$;
3. $TEM=tril(EM,-1)$; /* Building lower triangular matrix */
4. $SEM=sparse(TEM)$; /* Building sparse matrix */
5. (T, E)=$graphminspantree(SEM)$; /* Constructing MST */
6. for $i$=1:$c$-1 do
7.     Finding $c$-1 largest edges in $T$; /* $T$ containing value of edge and its vertex index */
8. end for
9. for $i$=1:$c$ do
10.    Traversing edges in $T$ excluding $c$-1 largest edges;
11. end for
12. Returning the disconnected parts in $T$ as $c$ clusters;
End.
```

FIGURE 3. **The initial clustering algorithm for core-points.**

an edge-weighted connected undirected graph that connects all the vertices, without any cycles and with the minimum total edge weight. Prim's algorithm and Kruskal's algorithm are two common used methods to obtain MST. The initial clustering algorithm is showed in Fig.3. We use Euclidean distance as the weight of edge. To save storage, sparse matrix is built based on low triangular matrix. After MST obtained, the $c-1$ largest weighted edges in MST are removed, and the c disconnected parts are returned as initial clustering.

C. ORDERING MECHANISM

Intuitively, the order of classifying the non-core points into initial clusters can affect the final cluster shape. Therefore, we develop a priority sequence for non-core points before classifying. The ordering mechanism of non-core points considers not only the neighborhood density but also the neighborhood compactness.

Definition 4 (Neighborhood Compactness of a Point p): Given a dataset D and k, the neighborhood compactness of a point p is defined as:

$$NC_{IS}(p) = \frac{|IS|}{\sum\limits_{q\in IS} \|p-q\|}, \quad IS = kNN(p) \cup RkNN(p),\ p \in D \tag{4}$$

In equation (4), where IS represents the influence space of point p, i.e. the union of k-nearest neighborhood and reverse k-nearest neighborhood of p. The larger p has the value of $NC_{IS}(p)$, the more compact neighborhood p owns.

Definition 5 (The Ordering Mechanism for Non-Core Points): Given a non-core point set NCP, p and q belong to NCP, p has a priority to q for classifying to initial clusters if p satisfies that:

$$\begin{cases} ND(p) > ND(q), \text{ or} \\ ND(p) = ND(q) \wedge NC\,(p) > NC\,(q)\,, \quad p, q \in NCP \end{cases} \tag{5}$$

We emphasize the ordering mechanism for non-core points instead of how to compute the value of ND and NC. We give our ordering algorithm for non-core points in Fig.4.

```
Algorithm Ordering
Input: $NCP$, $ND_{NCP}$, $IS_{NCP}$ /* $NCP$ is the non-core points set, $ND_{NCP}$ is an array containing neighborhood density of non-core points, $IS_{NCP}$ is a matrix containing influence space of non-core points */
Output: $PQ_{NCP}$ /* $PQ_{NCP}$ is the priority sequence of non-core points */
Begin
1. Initializing $PQ_{NCP}=\emptyset$;
2. for each point $p\in NCP$ do
3.     Computing $NC_{IS}(p)$; /*according to definition 4 */
4. end for
5. for points $p, q\in NCP$ do
6.     Sorting $p, q$;
7.     if $ND_{NCP}(p)> ND_{NCP}(q)$
8.         $PQ_{NCP} = PQ_{NCP}\cup\{p\}$;
9.     else if $ND_{NCP}(p)= ND_{NCP}(q)$ and $NC_{IS}(p)> NC_{IS}(q)$
10.        $PQ_{NCP} = PQ_{NCP}\cup\{p\}$;
11.    else
12.        $PQ_{NCP} = PQ_{NCP}\cup\{q\}$;
13.    end if
14. end for
15. Returning $PQ_{NCP}$;
End.
```

FIGURE 4. **The ordering algorithm for non-core points.**

D. NEAREST NEIGHBOR CLASSIFIER

In this module, we classify the non-core points into initial clusters by the nearest neighbor similarity measure because it is efficient and popular in machine learning. Fig.5 shows the partitioning clustering algorithm.

E. TIME COMPLEXITY ANALYSIS

The time complexity of our algorithm is composed of four parts. In density partition, the time complexity of searching

```
Algorithm partitioning clustering
Input: initial c clusters {C_1, C_2, ... , C_c}, PQ_NCP with n points
Output: final c clusters

Begin
1. for i=1:n; do
2.     point p = PQ_NCP(i);
3.     Finding p's the nearest neighbor q in c clusters;
4.     Classifying p into the cluster which q belongs to;
5.     Updating c clusters;
6. end for
7. Returning final c clusters;
End.
```

FIGURE 5. The partition clustering algorithm.

kNN is O($n$log$n$) by k-d tree. For initial clustering part, supposing that the size of core points set is m, the time complexity is $O(m * m + m \log m + bm)$, where $m < n$ and b is a constant. For last two modules, the time complexity is the same as $O((n-m)^*m)$, where $(n-m)$ is the size of non-core points set. Overall, the worst time complexity of our algorithm is $O(n^2)$.

V. EXPERIMENTAL RESULTS

In this section, both synthetic and real-world datasets are employed to evaluate the performance of the proposed clustering algorithm.

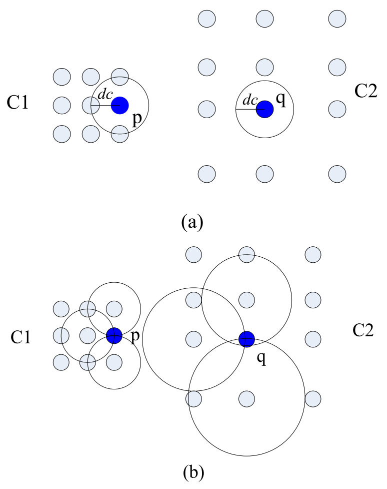


FIGURE 6. The comparison of *dc*-neighborhood and reverse *k* nearest neighborhood. (a) *dc*-neighborhood. (b) reverse *k* nearest neighborhood.

A. PERFORMANCE OF NEIGHBORHOOD DENSITY ESTIMATION

Generally, we use distance neighborhood to estimate the local density. However, it needs to set the minimum number of data points like DBSCAN and Dp algorithms. In Fig.6(a), there is an example of distance neighborhood. C1 and C2 are two clusters, data point p belongs to C1 and point q belongs to C2. We note that p has 3 neighbors and q has 0 neighbor when the distance is set to dc. To recognize cluster C2, we should increase the value of dc or set the minimum number of data points in dc neighborhood. Fig.6(b) shows an example of reverse k nearest neighborhood when $k = 2$. We can see that point p in a dense region of C1 has 3 reverse nearest neighbors, and point q as a core point of C2 also has 3 reverse nearest neighbors though it is in a sparse area. In other word, reverse nearest neighbors of a point p not only convey the distances between p and the surrounding data points, but also reflect the number of data points around p. In practice, it is easy to compute the number of RkNN.

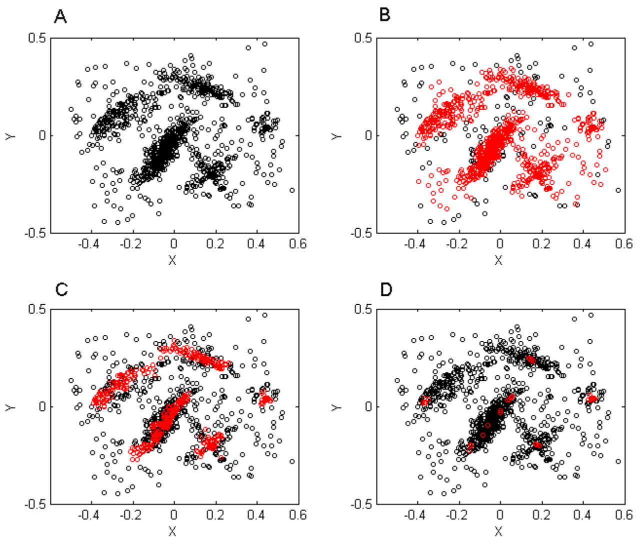


FIGURE 7. The quartile of local density of a synthetic 2-*d* dataset.

We conduct an experiment to evaluate the performance of the local density estimation model. The experimental dataset has 2000 samples embedded in 2-d space [4] and it is showed in Fig.7A. In experiments, we compute the local density of dataset. The 1/4 quartile of local density is showed in Fig.7B, where the points with red color mean that their local density are higher than other points. In a similar manner, the 1/2 and 3/4 quartile of local density are given in Fig.7C and Fig.7D. Especially, we see that half of dataset points with higher local density can clearly describe the cluster shape in Fig.7B. In Fig.7D, the 1/4 data points with higher local density can identify the cluster structure more accurately, even if the size of the cluster is smaller than that of other clusters, besides, for the cluster with oblong shape, red points are properly scattered across the cluster. In conclusion, the higher local density a point has, the closer it is away from the cluster center or dense region. The core points can represent cluster structure.

B. THE CHOICE OF PARAMETER k

In this section, we discuss the choice of parameter k influenced on the number of RkNN. The test dataset is showed in Fig.8A including 152 data points. It has a 'V' shape distribution, the center is dense region and the two ends are relatively sparse regions. As shown in Fig.8B, the number of RkNN is increased when k takes different values $k = 3, 7, 11, 15$. It is important that the shape of RkNN curves keeps basically the

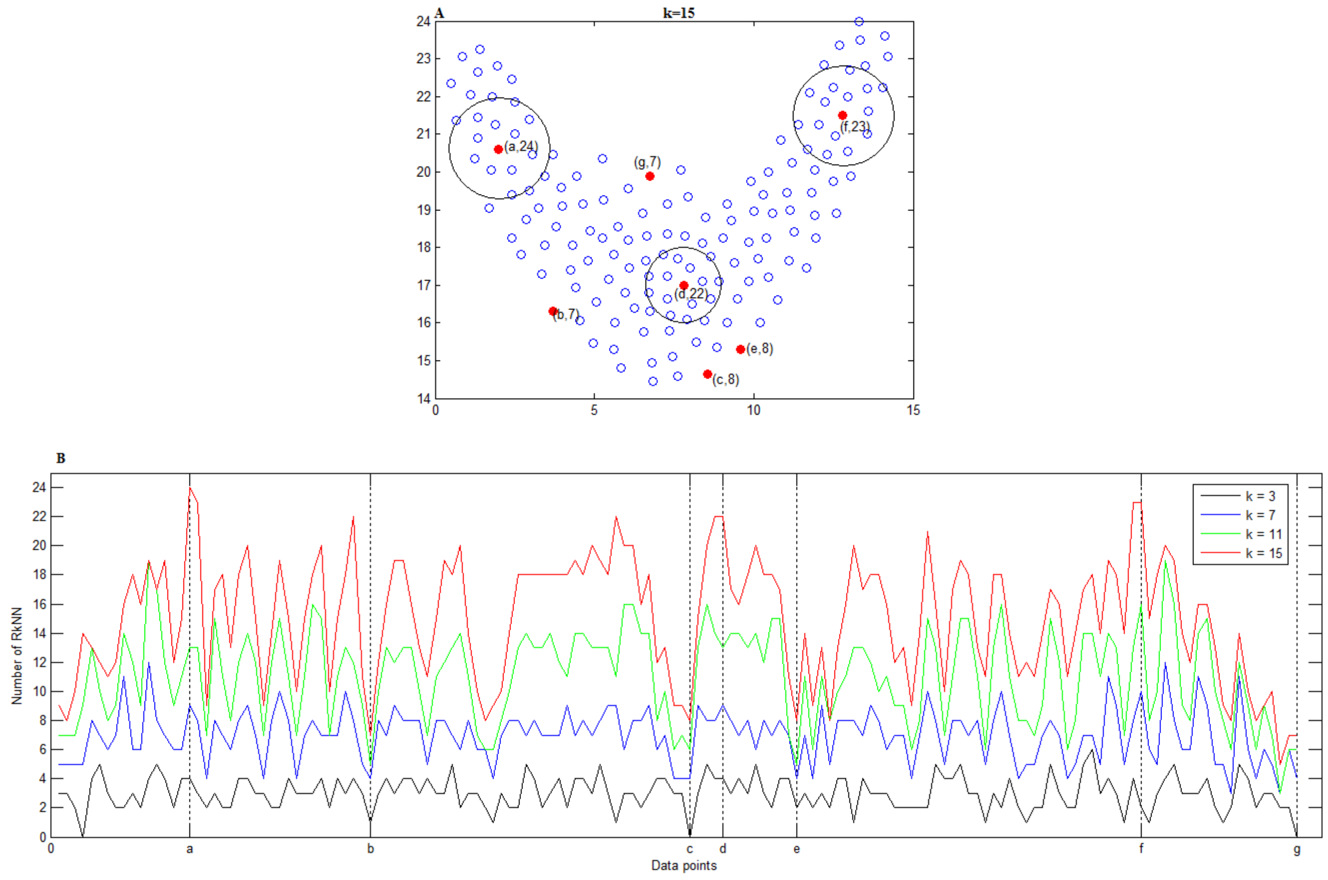


FIGURE 8. The choice of parameter k influenced on the number of RkNN.

same. It illustrates that the increase of the number of RkNN is relatively stable rather than irregular. More specifically, in Fig.8B, the number of RkNN of the data point a is 4, 9, 13, 24 when $k = 3, 7, 11, 15$, respectively. We denote data points in Fig.8B by

$$\begin{aligned}
&a(R3NN = 4,\ R7NN = 9,\ R11NN = 13,\ R15NN = 24),\\
&b(R3NN = 1,\ R7NN = 4,\ R11NN = 5,\ R15NN = 7),\\
&c(R3NN = 0,\ R7NN = 4,\ R11NN = 6,\ R15NN = 8),\\
&d(R3NN = 4,\ R7NN = 9,\ R11NN = 13,\ R15NN=22),\\
&e(R3NN = 2,\ R7NN = 4,\ R11NN = 5,\ R15NN = 8),\\
&f(R3NN = 2,\ R7NN = 10,\ R11NN = 16,\ R15NN=23),\\
&g(R3NN = 0,\ R7NN = 4,\ R11NN = 6, R15NN = 7).
\end{aligned}$$

We can find that the numbers of RkNN of the points a, d and f are increased rapidly but the numbers of RkNN of the points b, c, e and g are increased slowly. From Fig.8A, the points a and f are in relatively dense regions, d is in a very dense region, b, c, e and g are in sparse regions. In Fig.8A, we can see that the points a and f have more RkNNs than the point d when $k = 15$. The reason is that it needs a larger dc neighborhood to find 15 nearest neighbors for the points in sparse regions. Thus, the points a and f appear in more points' neighborhood and have more RkNNs.

C. PERFORMANCE ON OUTLIERS DETECTION

In this section, we conduct an experiment to evaluate the performance of our neighborhood density estimation method for outlier detection. The experiment dataset contains two clusters C1 and C2 with different sizes and densities, two local outliers p and q, a global outlier r and an outlying cluster C3 (see Fig.9(a)).

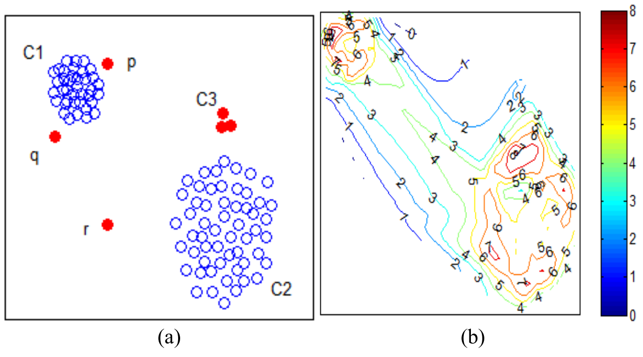


FIGURE 9. (a) A synthetic dataset contained outliers. (b) Contour map of the number of RkNNs.

Fig.9(b) shows the dataset's contour map of the number of RkNNs when k=8. We can see that the outliers p, q and r

TABLE 1. Overview of the synthetic datasets used in experiments.

Datasets	No. Samples	Dimensions	No. Clusters	Characteristics of datasets
Compound [25]	399	2	6	different shape, size and density
D31 [26]	3100	2	31	different overlaps
DS3 [14]	8000	2	6	noises, different shape, size and density
DS4 [14]	8000	2	9	noises, different shape, size and density
Flame [27]	240	2	2	noises, different shape, overlaps

have the least number of RkNNs than other points. The points in C3 own less number of RkNNs than the points in C1 and C2. Furthermore, the border points of C1 and C2 usually have less number of RkNNs than that of the inner points of C1 and C2. In our initial clustering, the points with more number of RkNNs are kept and other points (e.g. noises, outliers, border points) are removed temporarily. Therefore, our clustering algorithm is robust to the outliers.

D. PERFORMANCE ON SYNTHETIC DATASETS

In this section, we compare our algorithm with Dp and DBSCAN clustering algorithms on synthetic datasets. The detailed descriptions of synthetic datasets are given in Table 1. The explanation of experimental setup for the cluster algorithms is as follows.

DBSCAN: In DBSCAN algorithm [3], it requires user to set two parameters *MinPts* and *Eps*. *MinPts* means the minimum number of points in *Eps* neighborhood. In experiments, we set the *MinPts* from 5 to 15, and adjust the *Eps* to find the suitable number of clusters.

Dp: In Dp algorithm [4], it needs user to choose the cluster centers (i.e. density peaks on the decision graph). However, in some cases, it is difficult to assign density peaks by the decision graph. In experiments, the vector of γ [4] is used to replace the decision graph, and the top C values of γ are chosen as cluster centers. C is the real number of clusters.

Our Method: There are two parameters k and C in our algorithm. K is the number of nearest neighbors and C is the real clusters number. In density partition module, it does not need user to set the density threshold T, we take median value of local density as the density threshold.

The clustering results on synthetic datasets of the three clustering algorithms are showed in Table 2. The first row shows clustering algorithms, and the first column shows datasets name. From Table 2, DBSCAN algorithm can detect noises and identify the different cluster shapes (e.g. datasets Compound and DS3). However, it is sensitive to overlaps and density variation of a cluster (e.g. datasets D31 and DS4). If *Eps* is too small, one cluster will be separated into two clusters. If *Eps* is too big, two clusters will be merged into one cluster. For Dp algorithm, it works well on the clusters with convex shapes (e.g. dataset D31) but it cannot identify the clusters with non-convex shapes (e.g. datasets DS3 and DS4). We note that representing cluster structure by one point cannot capture the cluster shape. In our algorithm, cluster is represented by a set of core points which have high neighborhood density and can describe the cluster structure. Thus, our algorithm achieves better performance on synthetic datasets. The advantages of our method are as follows: 1) The number of RkNN is more effective to detect noises, border and dense points; 2) Clustering core points firstly eliminates the effects of cluster overlaps and density variation; 3) Classifying non-core points into initial clusters in sequence without influence on the final cluster shape.

E. PERFORMANCE ON REAL-WORLD DATASETS

In this section, we conduct experiments on real datasets [28] and use two cluster evaluation indexes *F1* [29] and rand index (*RI*) [30] to compare our algorithm with DBSCAN and Dp algorithm. The explanations of *F1* and *RI* are as follows.

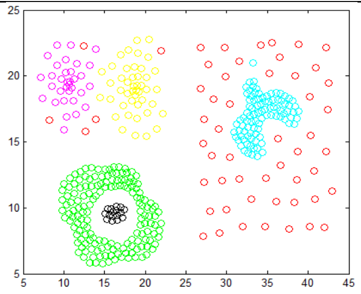
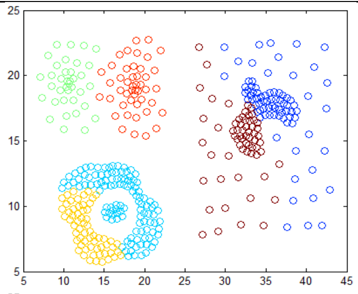
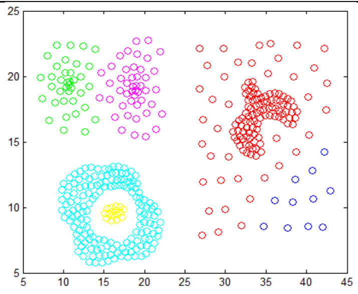
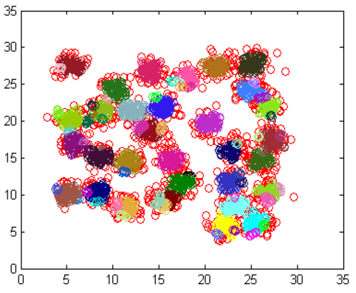
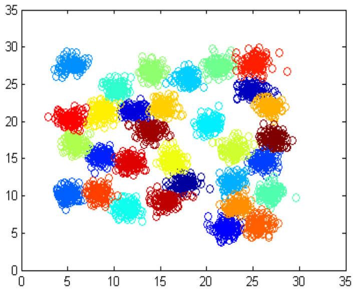
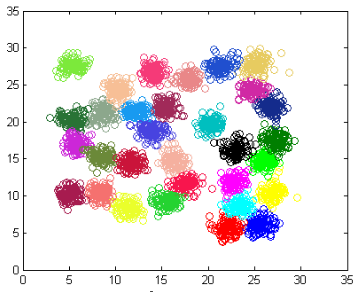
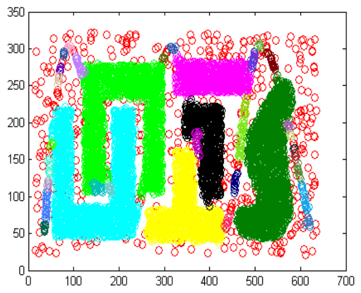
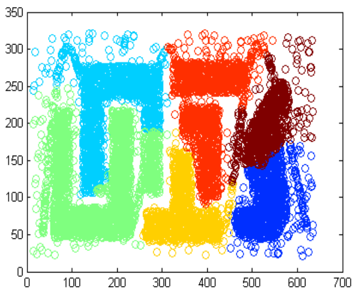
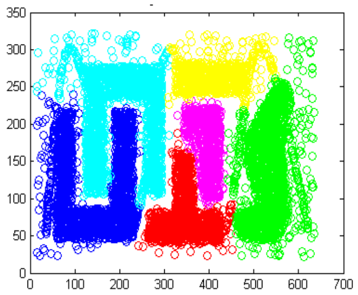
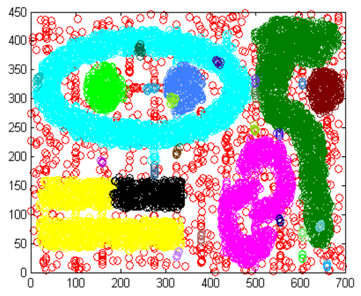
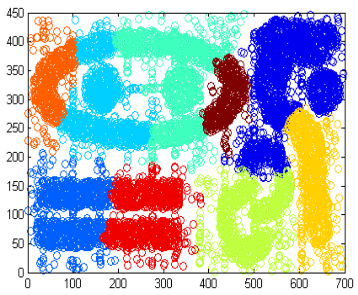
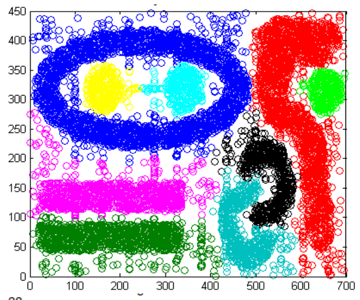
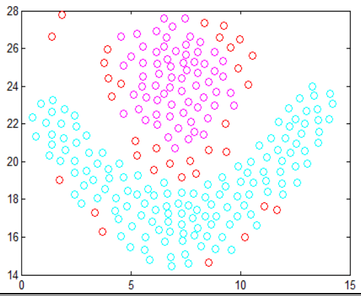
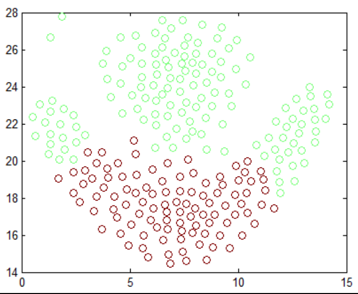
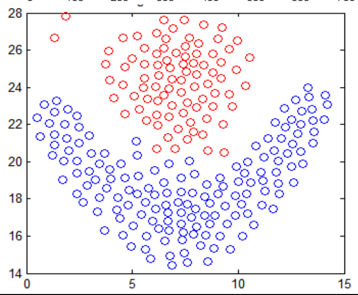
$$F1 = \frac{2 * P * R}{P + R} \tag{6}$$

$$RI = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

In equation (6), where $P = TP/(TP+FP)$, $R = TP/(TP+FN)$. *TP* is the number of true positives, *TN* is the number of true negatives, *FP* is the number of false positives, and *FN* is the number of false negatives. Both *F1* and *RI* take on a value between 0 and 1. The larger the values are, the better the clustering performance. The experimental results are showed in Table 3.

From Table 3, DBSCAN achieves the best results on control dataset. It obtains the highest F1 value and comparable RI value on sonar dataset. It is because that there exist obvious changes of density in control and sonar datasets. Besides, since cancer dataset has no significant variation of density, Dp is easy to find the density peaks and achieves the best performance on cancer dataset. We see that our method achieves better performance than Dp algorithm through all datasets except cancer dataset. For both *F1* and *RI*, our method exceeds Dp algorithm on four datasets and obtains

TABLE 2. Compare DBSCAN and Dp with our algorithm on synthetic datasets.

Datasets	DBSCAN	Dp	Our method
Compound			
D31			
DS3			
DS4			
Flame			

almost the same results of Dp algorithm on seeds dataset. About wine dataset, our method is not as good as Dp on *RI* but it is better than Dp on *F1*. Compared with DBSCAN, our method makes better results on five datasets except control dataset, and achieves comparative results on sonar and control datasets. Overall, the proposed algorithm is an effective clustering algorithm and achieves better or comparable results with state of the art clustering algorithms.

TABLE 3. Compare DBSCAN and Dp with our algorithm on *UCI* datasets.

Datasets	Dimensions	No. Samples	No. Clusters	*F1*			*RI*		
				DBSCAN	Dp	Our method	DBSCAN	Dp	Our method
cancer	30	569	2	0.65	**0.74**	0.69	0.51	**0.67**	0.55
control	60	600	6	**0.70**	0.58	0.67	**0.87**	0.84	0.86
diabetes	8	768	2	0.66	0.52	**0.70**	**0.55**	0.50	**0.55**
iris	4	150	3	0.82	0.84	**0.91**	0.88	0.89	**0.94**
seeds	7	199	3	0.62	**0.81**	0.80	0.71	**0.87**	**0.87**
sonar	60	208	2	**0.67**	0.51	0.65	0.50	0.50	**0.51**
wine	13	178	3	0.51	0.58	**0.63**	0.62	**0.72**	0.65

F. PERFORMANCE ON OLIVERTTI FACE DATABASE

We also applied our method to Olivetti Face Database [31] which is a public and popular benchmark for machine learning algorithms. In experiment, the similarity between two images was computed by the measure proposed in [32], we compared our method with Dp algorithm for the first 100 images of Olivetti Face Database, and cited the clustering results of Dp algorithm [4] in Fig.10. In the figure, faces with the same color belong to the same cluster, whereas gray images are not assigned to any clusters. Cluster centers are labeled with white circles. The clustering results of our method are showed in Fig.11, where digit on face represents cluster label, faces with the same digit belong to the same cluster, faces with green digit are correct clustering, and faces with red digit are wrong clustering.

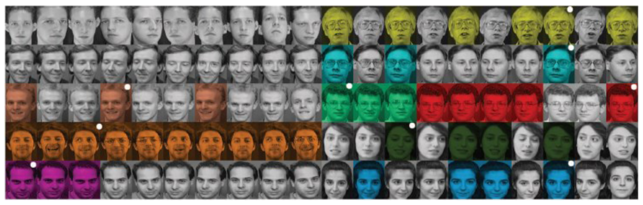


FIGURE 10. Clustering results of Dp algorithm on Olivetti Face Database for the first 100 images. Faces with the same color belong to the same cluster, whereas gray images are not assigned to any clusters. Cluster centers are labeled with white circles.

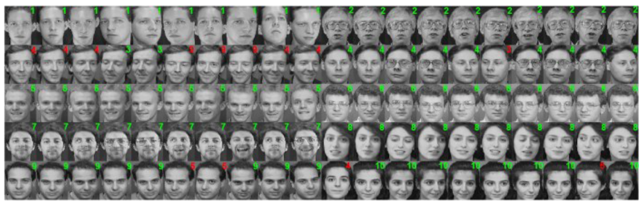


FIGURE 11. Clustering results of our method on Olivetti Face Database for the first 100 images. Digit on face represents cluster label, faces with the same digit belong to the same cluster, faces with green digit are correct clustering, and faces with red digit are wrong clustering.

From the results, Dp algorithm failed to identify 10 cluster centers. Besides, there are two centers in the same cluster in Fig.10. On the other hand, Dp algorithm has an unsatisfactory clustering accuracy because a small part of face images of each subject is correct clustering. Owing to the variation of facial expression and posture, face images have different structures. In consequence, it is inappropriate to use a face image to represent one subject. Our method achieves better performance in Fig.11, where 10 clusters are recognized correctly. We see that 5 subjects (i.e., 1st, 2nd, 6th, 7th, 8th subjects) can be recognized unambiguously and the 10th cluster is pure. For the remaining 3rd, 4th, 5th, 9th clusters, the clustering sizes are 3, 15, 13, 11, and 2, 9, 10, 8 images are correct clustering, respectively.

TABLE 4. Compare Dp with our algorithm on Olivetti face database.

Olivetti Face Database	No. clusters	*F1*		*RI*	
		Dp	Our method	Dp	Our method
DS1	10	0.61	**0.80**	0.92	**0.96**
DS2	20	0.48	**0.55**	0.93	**0.95**
DS3	30	0.45	**0.46**	0.95	**0.96**
DS4	40	0.38	**0.39**	**0.96**	**0.96**

In addition, we compare Dp algorithm with our method on the whole Olivetti Face Database, and the results are showed in Table 4. We produced four databases from Olivetti Face Database, which are DS1 (i.e. the first 100 images), DS2 (i.e. the first 200 images), DS3 (i.e. the first 300 images), and DS4 (i.e. the total 400 images), respectively. Since it has high similarity between faces, we adjust the density threshold T to the upper quartile of ND in density partition module. From the Table 4, the performance of our method is better than Dp algorithm especially on the DS1 and DS2. Overall, for our method, *F1* takes on a value 0.8 when the number of clusters is 10, but the value of *F1* drops to 0.55 when the clusters number reaches to 20. With the increasing of clusters number, the value of *F1* keeps dropping down. However, there is almost no change about the value of *RI*. The reason is that *RI* is highly dependent upon the number of clusters, and *RI* converges to 1 as the number of clusters increases [33].

VI. CONCLUSION

In this paper, we proposed a simple but effective clustering algorithm based on density clustering framework, which can combine the advantages of different clustering algorithms to obtain the desired clustering results. In experiments, we evaluate clustering performance of the proposed algorithm on synthetic and real-world datasets. The experimental results show that our method is effective and achieves comparable performance with DBSCAN and Dp algorithms. Our method reveals that core points are better to represent cluster structure. In the future work, we will use the existing clustering techniques (e.g., k-means, DBSCAN) to fit into the framework to improve their performance and compare DCF with more other benchmark methods. Furthermore, we will study the selection of the parameter k in neighborhood estimation model from the theoretical aspect.

REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Aug. 2010.

[2] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[3] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.

[4] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[5] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[6] R. Scitovski and K. Sabo, "Analysis of the k-means algorithm in the case of data points occurring on the border of two or more clusters," *Knowl.-Based Syst.*, vol. 57, pp. 1–7, Feb. 2014.

[7] G. Tzortzis and A. Likas, "The MinMax k-means clustering algorithm," *Pattern Recognit.*, vol. 47, pp. 2505–2516, Jul. 2014.

[8] S. Eschrich, J. Ke, L. O. Hall, and D. B. Goldgof, "Fast accurate fuzzy clustering through data reduction," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 2, pp. 262–270, Apr. 2003.

[9] L. Hubert, "Approximate evaluation techniques for the single-link and complete-link hierarchical clustering procedures," *J. Amer. Statist. Assoc.*, vol. 69, no. 347, pp. 698–704, 2012.

[10] J. Xu, G. Wang, and W. Deng, "DenPEHC: Density peak based efficient hierarchical clustering," *Inf. Sci.*, vol. 373, pp. 200–218, Dec. 2016.

[11] S. Zhou, Y. Zhao, J. Guan, and J. Huang, "A neighborhood-based clustering algorithm," in *Advances in Knowledge Discovery and Data Mining*, vol. 3518. Berlin, Germany: Springer, Nov. 2005, pp. 361–371.

[12] Y. Zhu, K. M. Ting, and M. J. Carman, "Density-ratio based clustering for discovering clusters with varying densities," *Pattern Recognit.*, vol. 60, pp. 983–997, Dec. 2016.

[13] O. Grygorash Y. Zhou, and Z. Jorgensen, "Minimum spanning tree based clustering algorithms," in *Proc. ICTAI*, 2006, pp. 73–81.

[14] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[15] G. Gan, Y. Zhang, and D. K. Dey, "Clustering by propagating probabilities between data points," *Appl. Soft Comput.*, vol. 41, pp. 390–399, Apr. 2016.

[16] G. Celeux and G. Govaert, "Gaussian parsimonious clustering models," *Pattern Recognit.*, vol. 28, no. 5, pp. 781–793, 1995.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[19] X. F. Wang and D. S. Huang, "A novel density-based clustering framework by using level set method," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1515–1531, Nov. 2009.

[20] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[21] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2006, pp. 577–593.

[22] L. Liu *et al.*, "An influence power-based clustering approach with PageRank-like model," *Appl. Soft Comput.*, vol. 40, pp. 17–32, Mar. 2016.

[23] L. Ertöz, M. Steinbach, and V. Kumar, "A new shared nearest neighbor clustering algorithm and its applications," in *Proc. Workshop Clustering High Dimensional Data Appl. 2nd SIAM Int. Conf. Data Mining*, 2002, pp. 105–115.

[24] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 201–212, 2000.

[25] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. 20, no. 1, pp. 68–86, Jan. 1971.

[26] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1273–1280, Sep. 2002.

[27] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinform.*, vol. 8, no. 1, p. 3, Jan. 2007.

[28] M. Lichman. (2013). Machine learning repository. University of California, Irvine, School of Information and Computer Sciences. [Online]. Available: http://archive.ics.uci.edu/ml

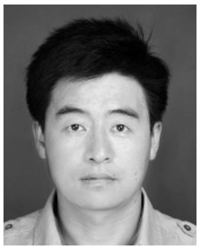
[29] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proc. KDD*, 1999, pp. 16–22.

[30] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Statist. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.

[31] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. IEEE Workshop Appl. Comput. Vis.*, Sarasota, FL, USA, Dec. 1994, pp. 138–142.

[32] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, "Complex wavelet structural similarity: A new image similarity index," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–2401, Nov. 2009.

[33] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Statist. Assoc.*, vol. 78, no. 383, pp. 553–569, 1983.



JIANYUN LU (M'16) received the M.S. degree in computer science from Chongqing University, China, 2010, where he is currently pursuing the Ph.D. degree with the College of Computer Science. Since 2012, he has been a Lecturer with the Chongqing College of Electronic and Engineering, China. His research interests are in data mining, machine learning, and big data.

QINGSHENG ZHU (M'11) received the B.S., M.S., and Ph.D. degrees in computer science from Chongqing University in 1983, 1986, and 1990, respectively. He is currently a Professor with the College of Computer Science, Chongqing University, and also the Director of the Chongqing Key Laboratory of Software Theory and Technology. His main research interests include Ecommerce, data mining, and service oriented computing.